Initial Performance Results of Nalu on aarch64

P3HPC, submission #48

(poster)

Srinath Vadlamani, Arm Inc.

Sept. 2nd, 2020 online recorded presentation

Motivation: How well does Arm architecture and μ-architecture work with HPC applications?

- A goal we have is to understand a true production code’s behavior on emergent architecture such as *aarch64.*
- Nalu is a SNL production code.
  - It serves a mission critical role .
  - It shares common code and infrastructure with export-controlled codes such as Sparc.
    - Sparc is not deployable on aarch64 systems such as AWS E2C Gravitons.
- Analyzing Nalu’s performance can benefit other DOE/NNSA mission critical codes as well.

Presentation outline.

- Review ideas for fair performance comparisons.
- Review Nalu code and experiment parameters.
- Show initials results
  - μ–architecture (ThunderX2, Graviton 2, x86s) simulation comparisons
  - Scaling result
- Propose plan to diagnose the performance differences across μ-architectures.
- State conclusions and future work.

Fair performance comparisons sufficient conditions.

- Software stack provenance helps the “apples to apples” comparison.
  - Software versions and build structures should be similar across the μ-architectures.
- Also, building and running the same cases (code + initial conditions) across the μ-architectures needs to be manageable.
  - Productivity is to be considered with both the many variants of builds and cases that are necessary to get insight into the different facets of μ-architectural performance.
    - Compilers + versions of software + build flags + etc.
    - Ex: implementations of MPI and its variants such as using UCX or Shmem.

- Spack (https://spack.readthedocs.io) can satisfy these conditions.
  - For these experiments:
    - the same *hash* develop branch was used across all platforms.
    - we used a simple command: *spack install nalu%gcc@10.2.0 ^openmpi@4.0.4 ^ trilinos~matio*
      - <c,cxx,f>flags = “-g1 –mtune=<μ-arch> Wno-error=coverage-mismatch”
        - Spack internal optimization for specific x86 platforms was used.

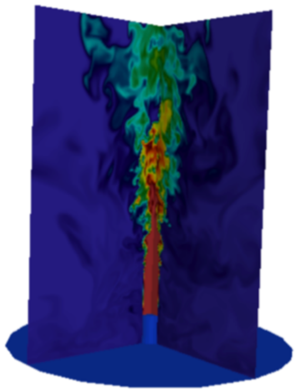
Review of Nalu: Sandia HPC mission critical code

What is Nalu?

- Low-mach number CFD code from SNL [1].
- Shares codes exercised in Sierra Framework [2].
  - In common with Sparc and mission critical codes
- Dependencies such as HDF5, PNetCDF, Trilinos [3,4], and Kokkos.
- Massively parallel using unstructured meshes
- Variable density turbulent flow capability designed for energy applications.
- Used in aerodynamic prediction and traditional gas-phase combustion simulation campaigns.

Experiment Parameters

- Miletsone case meshR2_nt.g
- mixture-fraction based turbulent open jet with a Reynolds number ~6000 emitting from base
- Meshes range 200 thousand to 9 trillion elements
  - 2 million elements for this case
- Half of unknowns in momentum solve
  - Rest split over elliptical poisson pressure (continuity) system, mixed-fraction, and kinetic energy systems
- Trilinos is the primary package for solving the systems while using Kokkos for data and memory management among the hierarchy of memory systems on the compute nodes.



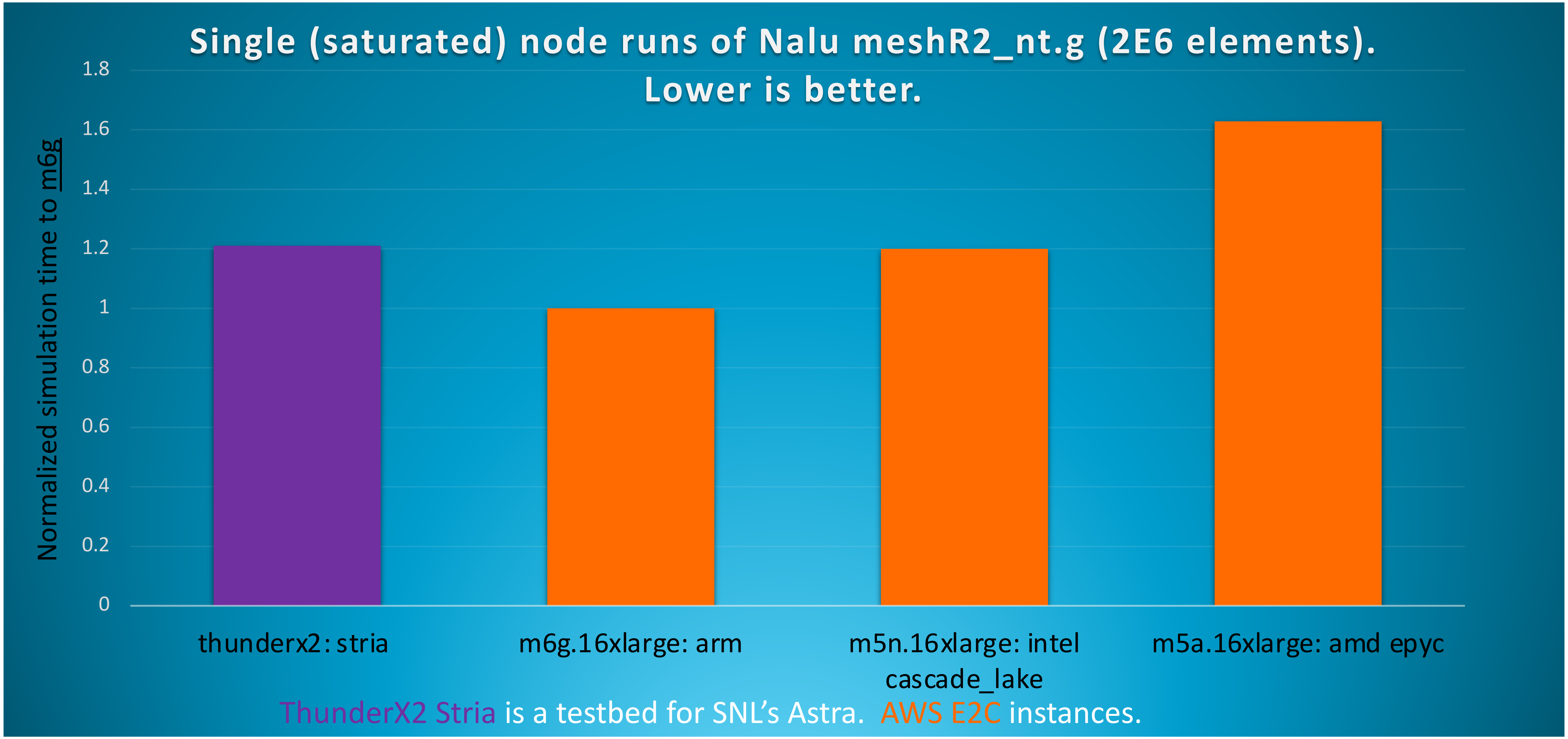
Fig[1]. Jet flow with initial condition in meshR2_nt.g milestone case

System attributes of HPC systems for performance comparisons

Instance	ThunderX2: Stria at SNL (Arm)	AWS M6g.16xlarge (Arm Graviton2)	AWS M5n.16xlarge: Intel	AWS M5a.16xlarge: AMD
Threads/core	2	1	2	2
Cores/socket	28	64	16	32
Sockets/node	2	1	2	1 with 4 NUMA regions
CPU GHz	2.0	2.5	3.2	2.3
Arch	ArmV8.1	ArmV8.2	X86-64 + AVX512	X86-64 + AVX2
µarch	ThunderX2	Neoverse N1	Cascade Lake	Zen
L1i, L1d cache (K)/core	32,32	64, 64	32, 32	32, 32
L2 cache (K)/core	256	1024	1024	512
L3 cache (K)/node	32768	32768	36608	65536
Mem. channels	8x DDR4-2666 per socket	8x DDR4-3200	6x DDR4-2933 per socket	8x DDR-2666, (2x per NUMA node)
TDP (Watts)	Approximate 235 x 2 sockets	Unavailable*	210 x 2 sockets	180 x 2 sockets

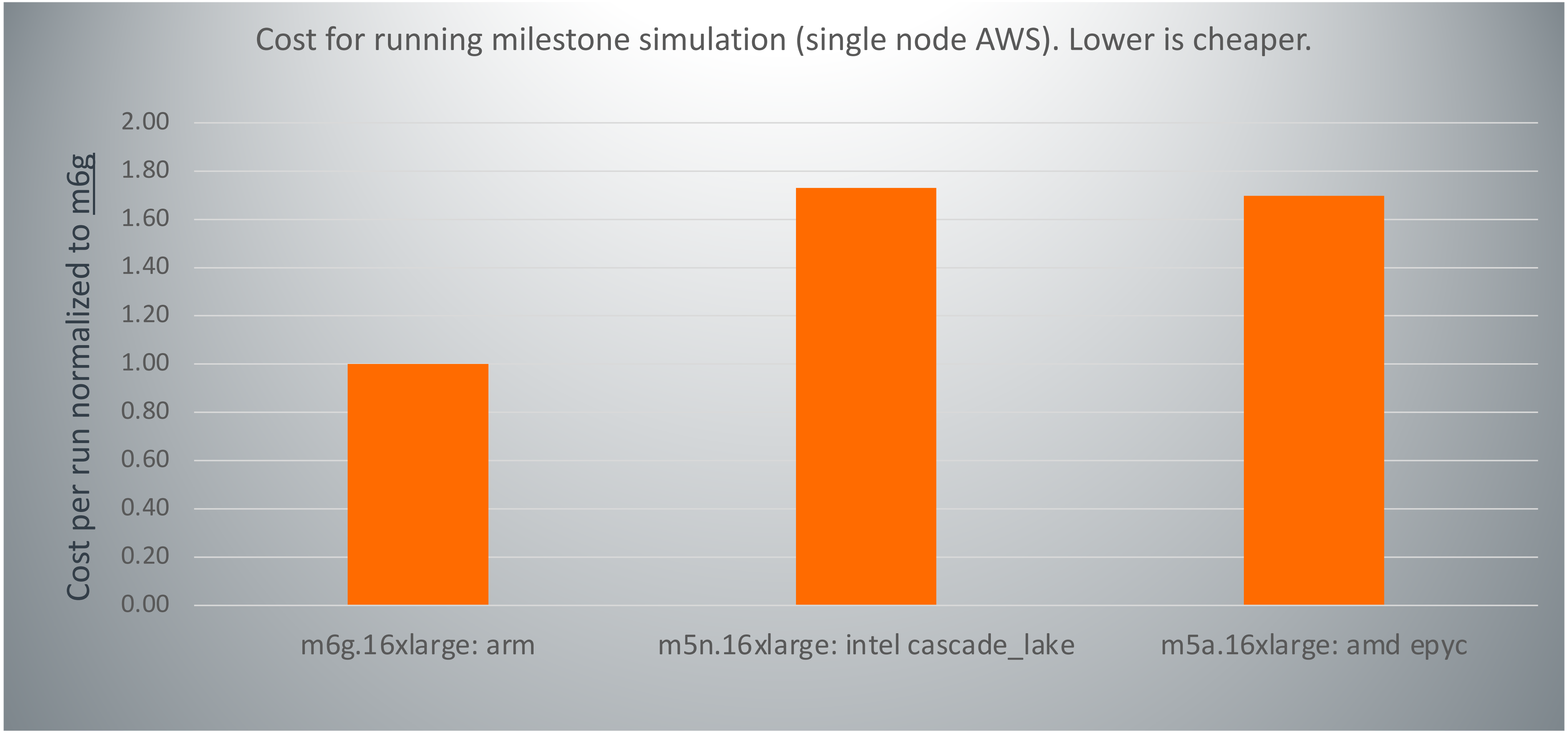
* Similarly sized N1 systems from Ampere Altra at 2.6GHz have TDP around **125W per socket, see link below**
https://www.servethehome.com/ampere-altra-max-targets-a-128-core-arm-cpu-shipping-in-2021/

Newer Arm-based systems show promising performance with baseline case.

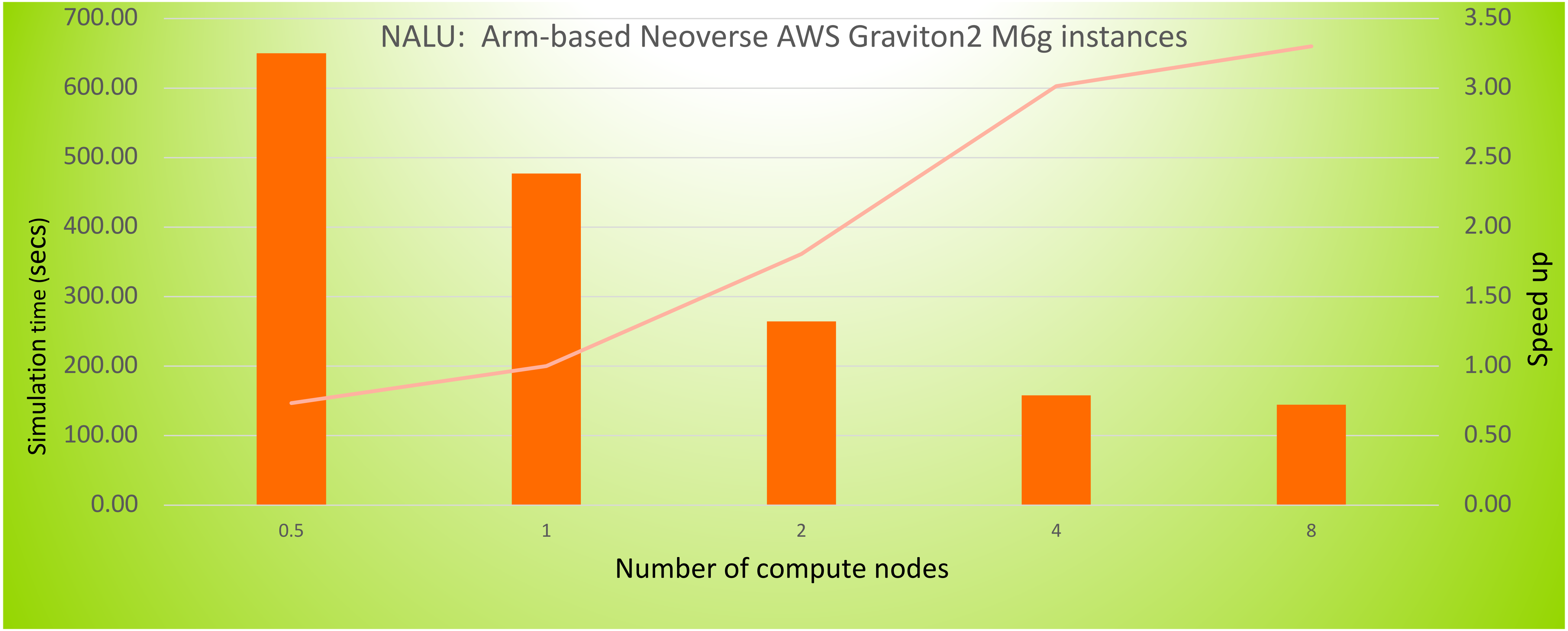


Compiler: gcc@10.2.0, Openmpi@4.0.4

Aarch64 is advantageous when considering simulation expense



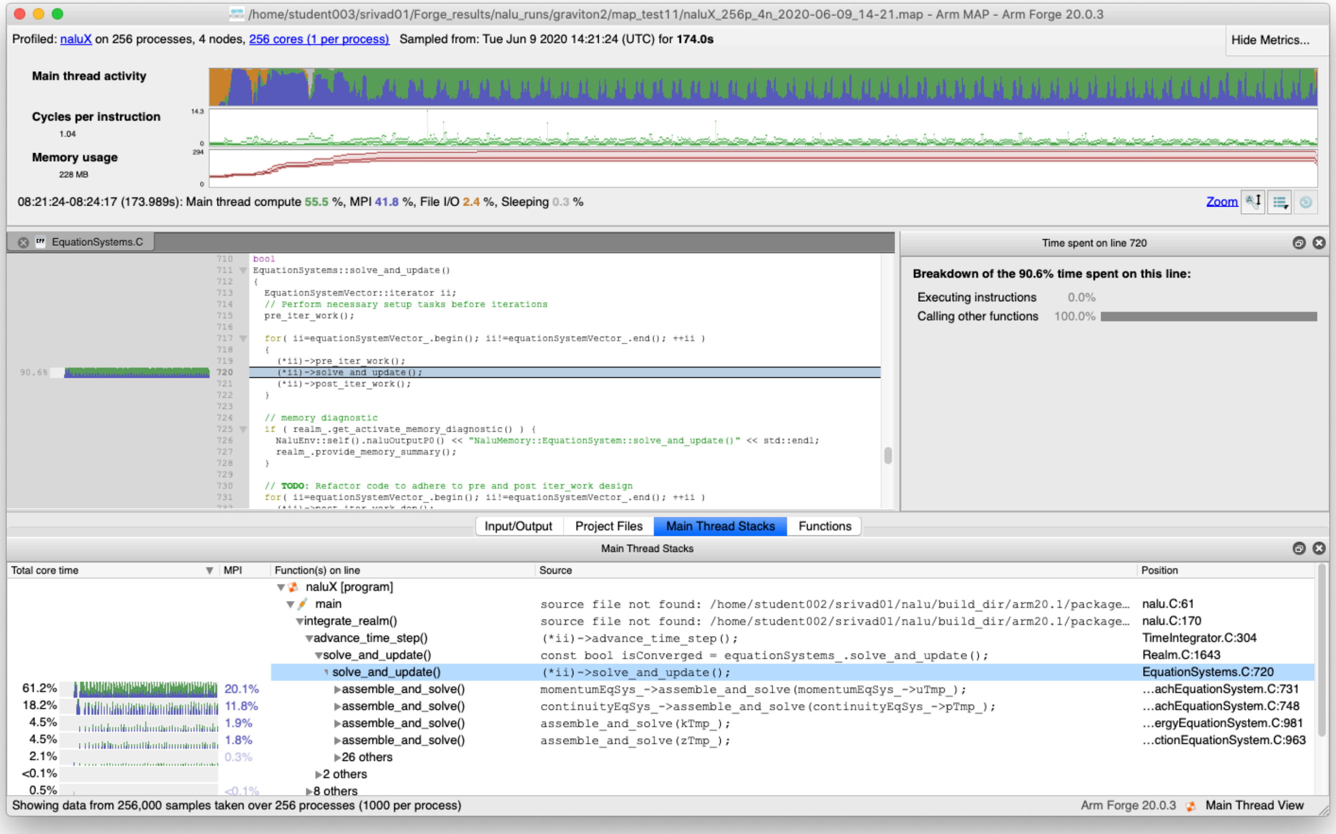
AWS parallel cluster can allow for paralle scaling.



This AWS parallel cluster instance had 8 nodes. A larger mesh (more elements) is required to exercise Nalu scaling capability at higher node count.

Strategies for pursing performance difference with profiling.

- Platform independent tools for profiling the cases can help explain differences across microarchitectures.
  - Cache misses and computing units usage are to be compared.
  - Frontend vs. backend stalls can give insight to candidates for bottlenecks.
- Tools of choice:
  - Arm Forge Map
    - Profiling with Perf and Papi
  - TAU
    - More extensive measurements are possible.



Ex. 1: 4nodes on M6g, best compute/data-movement ratio profiled behavior.

Conclusions:

- Newer Arm-based systems show promising performance with the baseline Nalu case.
  - Both in time to solution and cost.
- Software management tools such as Spack helps performance analysis across μ-architectures.
  - The results use same variants thusly reducing the degrees of freedom for initial comparisons: *baseline of configuration*.
  - Provenance of the software stack can be attained easily.
    - The same versions of the entire software stack for these comparisons are exactly the same, including compiler.
- Future work
  - More compilers that are greatly tuned for specific architecture.
  - A64FX and Ampere multimode systems and are being deployed
    - Offering a larger microarchitectural landscape with different memory subsystems and floating point engines.
  - Profiling will allow insight to the compute vs. data movement ratio of the microarchitectures.
    - Highlighting microarchitectures that are amenable to current and future HPC workloads.

References:

- [1] Domino, S. "Sierra Low Mach Module: Nalu Theory Manual 1.0", SAND2015-3107W, Sandia National Laboratories Unclassified Unlimited Release (UUR), 2015. https://github.com/NaluCFD/NaluDoc.
- [2] H. Edwards, A. Williams, G. Sjaardema, D. Baur, and W. Cochran. Sierra toolkit computational mesh computational model. Technical Report SAND-20101192, Sandia National Laboratories, Albuquerque, NM, 2010.
- [3] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, J. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An overview of trilinos. Technical Report SAND-20032927, Sandia National Laboratories, Albuquerque, NM, 2003.
- [4] S. Tieszen, S. Domino, and A. Black. Validation of a simple turbulence model suitable for closure of temporally-filtered navier-stokes equations using a helium plume. Technical Report SAND-20053210, Sandia National Laboratories, Albuquerque, NM, June 2005.
- [5] A. M. Agelastos, P. T. Lin. Simulation Information Regarding Sandia National Laboratories' Trinity Capability Improvement Metric. Sandia Report SAND2013-8748, Sandia National Laboratories, Albuquerque, NM, October 17, 2013
- [6] P. T. Lin, M. T. Bettencourt, S. Domino, T. Fisher, M. Hoemmen, J. J. Hu, E. T. Phipps, A. Prokopenko, S. Rajamanickam, C. M. Siefert and Stephen Kennon, Towards Extreme-Scale Simulations for Low Mach Fluids with Second-Generation Trilinos, Parallel Process. Lett., vol. 24, 2014.
- [7] Todd Gamblin, Matthew P. LeGendre, Michael R. Collette, Gregory L. Lee, Adam Moody, Bronis R. de Supinski, and W. Scott Futral. The Spack Package Manager: Bringing Order to HPC Software Chaos. In Supercomputing 2015 (SC'15), Austin, Texas, November 15-20 2015. LLNL-CONF-669890.

Thank you.